**AutoHealth - Health Record Management System**

This Python script provides functionalities for managing health records in an Excel worksheet, calculating BMI, sending quarterly reports with health information via email, and scheduling reminders for checkups.

**Please note:** This snippets remains a sample for my clients for demonstration purposes only and is not intended for production use. It lacks crucial functionalities like user authentication, data security, and compliance with healthcare regulations and I am trained to put those in consideration in my training as a software engineer. Omitting such was only to provide demonstration to clients with working snippets, to demonstrate advanced use of python language, its functions and libraries to tackle their needs however complex.

**Installation**

1. Make sure you have Python installed on your system.

2. Install the required libraries using `pip`:

```bash

pip install openpyxl smtplib

```

**Configuration**

1. Replace `"health\_records.xlsx"` with the actual filename of your Excel sheet containing health records comensurate to the snippets.

2. May choose to modify the email sender and recipient information in the `send\_email` function within the script.

3. Implement the logic and data source for retrieving health information relevant to each user in the `send\_quarterly\_report` function (currently a placeholder).

**Usage**

1. Save the code as `autohealth.py`.

2. Run the script from your terminal:

```bash

python autohealth.py

```

**This will**

1. Load health data from the Excel sheet.

2. Send quarterly reports and checkup reminders to users with email addresses in the data.

3. (Placeholder) Retrieve and include relevant health information in the reports.

**Functionalities**

load\_data(): Reads health data (name, weight, height, email) from the specified Excel sheet.

save\_data(data): Saves updated health data back to the Excel sheet.

calculate\_bmi(weight, height): Calculates the Body Mass Index (BMI) based on weight and height.

send\_email(email, report, subject): Sends an email with the provided report and subject to the recipient.

send\_quarterly\_report(data): Identifies users due for a quarterly report based on the current quarter and their last report date.

Generates a report with weight, height, BMI, and retrieves relevant health information (currently a placeholder).

Sends the report and a checkup reminder email to the user.

Updates the user's last report date in the data.

main(): Loads data, sends quarterly reports, and schedules the next report (currently a placeholder).

**Disclaimer**

This code is provided as a basic example by compusolvetechnologies@gmail.com and should not be used without proper modifications, security considerations, and adherence to relevant regulations. Building a healthcare application requires expertise in data security, privacy, and compliance which we ommited to provide this sample to you. Consult with us as professionals for secure and compliant implementation of this or a similar system or to carry out your project at an affordable price.

**Possible upgrade/improvements to this system**

Data Storage: You can use the library in the snippets "openpyxl" to interact with the Excel worksheet and manage health records. To upgrade, consider switching to a more scalable database solution like PostgreSQL in the future.

User Authentication: Implement a secure user authentication system using libraries like Django REST framework or third-party services like Firebase Authentication. They have inbuilt security and authentication features.

BMI Calculator: You can create a function to calculate the Body Mass Index (BMI) based on the user's weight and height stored in the health record to accommodate different values such as people input of body mass in lb or kg and height in cm, m, etc. This will ensure a more robust system adept to any data fed in regardless of unit. Hence, a consideration of flexible unit conversion is necessary.

Email Sending: We used libraries like smtplib, to upgrade, you could easily consider integrating with third-party email service providers like "SendGrid" or "Mailgun" to send health reports and notifications.

**Coding of AutoHealth**

import openpyxl

from datetime import datetime, timedelta

from email.mime.multipart import MIMEMultipart

from email.mime.text import MIMEText

import smtplib

def load\_data():

"""Loads health data from an Excel worksheet."""

wb = openpyxl.load\_workbook("health\_records.xlsx")

sheet = wb["Sheet1"]

data = []

for row in sheet.iter\_rows(min\_row=2):

data.append(

{

"name": row[0].value,

"weight": row[1].value,

"height": row[2].value,

"email": row[3].value,

}

)

return data

def save\_data(data):

"""Saves health data to an Excel worksheet."""

wb = openpyxl.load\_workbook("health\_records.xlsx")

sheet = wb["Sheet1"]

sheet.delete\_rows(2, sheet.max\_row) # Clear existing data

for i, row in enumerate(data, start=2):

sheet.cell(row=i, column=1).value = row["name"]

sheet.cell(row=i, column=2).value = row["weight"]

sheet.cell(row=i, column=3).value = row["height"]

sheet.cell(row=i, column=4).value = row["email"]

wb.save("health\_records.xlsx")

def calculate\_bmi(weight, height):

"""Calculates the Body Mass Index (BMI)."""

if height == 0:

return 0

return weight / (height / 100) \*\* 2

def send\_email(email, report, subject):

"""Sends an email with the health report."""

message = MIMEMultipart()

message["From"] = "sender@example.com"

message["To"] = email

message["Subject"] = subject

body = MIMEText(report, "plain")

message.attach(body)

with smtplib.SMTP("smtp.example.com", 587) as server:

server.starttls()

server.login("sender@example.com", "password")

server.sendmail(message["From"], message["To"], message.as\_string())

def send\_quarterly\_report(data):

"""Sends quarterly reports and checkup reminders."""

current\_quarter = datetime.now().quarter

for record in data:

if record["email"]:

if record.get("last\_report\_quarter") != current\_quarter:

report = f"Health Report for {record['name']}:\n"

report += f"\tWeight: {record['weight']} kg\n"

report += f"\tHeight: {record['height']} cm\n"

report += f"\tBMI: {calculate\_bmi(record['weight'], record['height']):.2f}\n"

# Add health information retrieval logic here (replace placeholder)

report += f"\n\*\*Health Information:\*\*\n"

report += f"\t[Placeholder for health information relevant to {record['name']}]"

send\_email(record["email"], report, "Quarterly Health Report")

record["last\_report\_quarter"] = current\_quarter

save\_data(data)

# Schedule next report for 3 months from now

next\_quarter = (current\_quarter + 3) % 4 or 12

# Replace with actual scheduling library/function

# schedule\_next\_report(next\_quarter)

def main():

data = load\_data()

send\_quarterly\_report(data)

if \_\_name\_\_ == "\_\_main\_\_":

main()